Name: Satyam Rai
UID: 2019130051
TE Comps Batch C

Exp6

Aim: To solve problems using Prolog Programming.

Q.1) Create a family tree using PROLOG. It should have rules for father, mother,brother, sister, grandparent, uncle, aunt, predecessors, successors.

Code:

```
parent(pandu, arjun).
parent(pandu, bheem).
parent(pandu, nakul).
parent(arjun, abhimanyu).
parent(arjun, pragati).
parent(abhimanyu, parikshit).

parent(uttara, parikshit).
parent(subhadra, abhimanyu).
parent(subhadra, pragati).
parent(kunti, arjun).
parent(kunti, bheem).
parent(madri, nakul).

female(kunti).
female(madri).
female(pragati).
female(uttara).
```

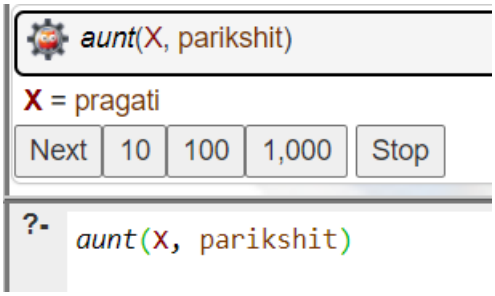
```
female(subhadra).

male(pandu).
male(arjun).
male(nakul).
male(abhimanyu).
male(parikshit).

mother(X, Y):- parent(X, Y), female(X).
father(X, Y):- parent(X, Y), male(X).

son(X, Y):- parent(Y, X), male(X).
daughter(X, Y):- parent(Y, X), female(X).

grandfather(X, Y):- parent(X, A), parent(A, Y), male(X).
grandmother(X, Y):- parent(X, A), parent(A, Y), female(X).

sister(X, Y):- parent(A, X), parent(A, Y), female(X), X \= Y.
brother(X, Y):- parent(A, X), parent(A, Y), male(X), X \= Y.

aunt(X, Y):- sister(X, Z), parent(Z, Y).
uncle(X, Y):- brother(X, Z), parent(Z, Y).

predecessor(X, Y) :- parent(X, Y), X\=Y.
predecessor(X, Y) :- parent(X, A),predecessor(A,
Y).
```

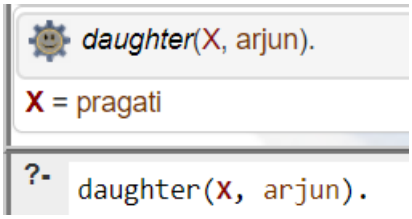
successor(X, Y):- son(Y, X), X\=Y.

successor(X, Y):- daughter(Y, X).

successor(X, Y):- son(A, X), successor(A, Y).

successor(X, Y):- daughter(A, X), successor(A, Y).





grandfather(X, parikshit)

X = arjun

Next | 10 | 100 | 1,000 | Stop

```
?- grandfather(X, parikshit)
```

grandmother(X, parikshit)

X = subhadra

Next | 10 | 100 | 1,000 | Stop

```
?- grandmother(X, parikshit)
```

mother(X, abhimanyu).

X = subhadra

```
?- mother(X, abhimanyu).
```

sister(X, abhimanyu)

X = pragati

Next | 10 | 100 | 1,000 | Stop

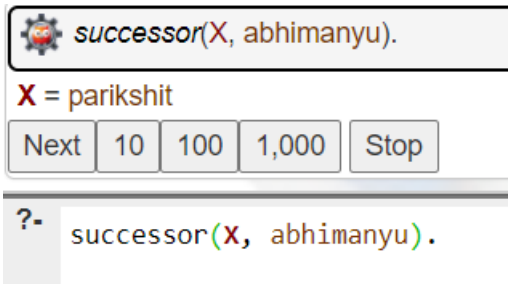
```
?- sister(X, abhimanyu)
```

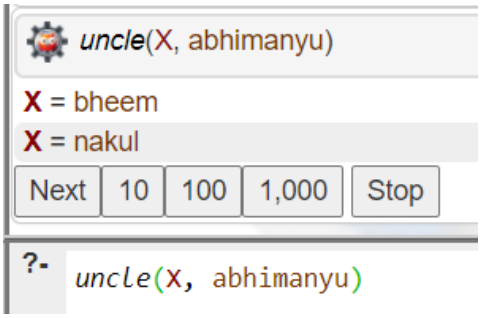
son(X, arjun)

X = abhimanyu

Next | 10 | 100 | 1,000 | Stop

```
?- son(X, arjun)
```

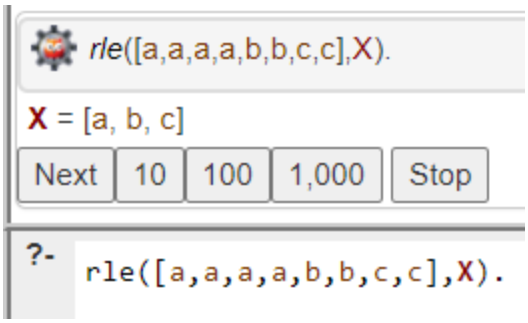




Q2)Given a list [a,a,a,a,b,b,b,c,c] write a function that does the following rle([a,a,a,a,b,b,c,c],X), X: [a,b,c]

Code:

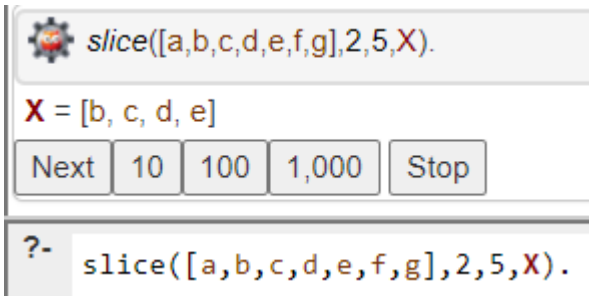
```
rle([],[]).
rle([X],[X]).rle([X, X|REMAINING],OUTPUT) :-
rle([X|REMAINING],OUTPUT).
rle([X, Y|REMAINING], [X|OUTPUT_TAIL]) :- X \= Y, rle([Y|REMAINING],
OUTPUT_TAIL).
```



Q3)Given a list [a,b,c,d,e,f,g] write a function that does the following
slice([a,b,c,d,e,f,g],[2,5],X), X: [c,d,e,f]

Code:

```
slice([X|_], 1, 1, [X]).
slice([X|TAIL], 1, CURRENT_INDEX, [X|REM_TAIL]) :- CURRENT_INDEX >
1,
  NEXT_INDEX is CURRENT_INDEX - 1, slice(TAIL, 1, NEXT_INDEX,
REM_TAIL).
slice([_|TAIL], I, CURRENT_INDEX, OUTPUT) :- I > 1,
  I1 is I - 1, NEXT_INDEX is CURRENT_INDEX - 1, slice(TAIL, I1,
NEXT_INDEX, OUTPUT).
```



Q4)

Group list into sublists according to the distribution givenFor

example,

subsets([a,b,c,d,e,f,g],[2,2,3],X,[]) should return X = [[a,b][c,d][e,f,g]]

The order of the list does not matter

Code:

```
el(X,[X|L],L).
 el(X,[_|L],R) :- el(X,L,R).

selectN(0,_,[]) :- !.
selectN(N,L,[X|S]) :- N > 0, el(X,L,R),
   N1 is N-1,
   selectN(N1,R,S).

subsets([],[],[],[]).
subsets(G,[N1|Ns],[G1|Gs],[]) :-
   selectN(N1,G,G1),
   subtract(G,G1,R),
   subsets(R,Ns,Gs,[]).
```

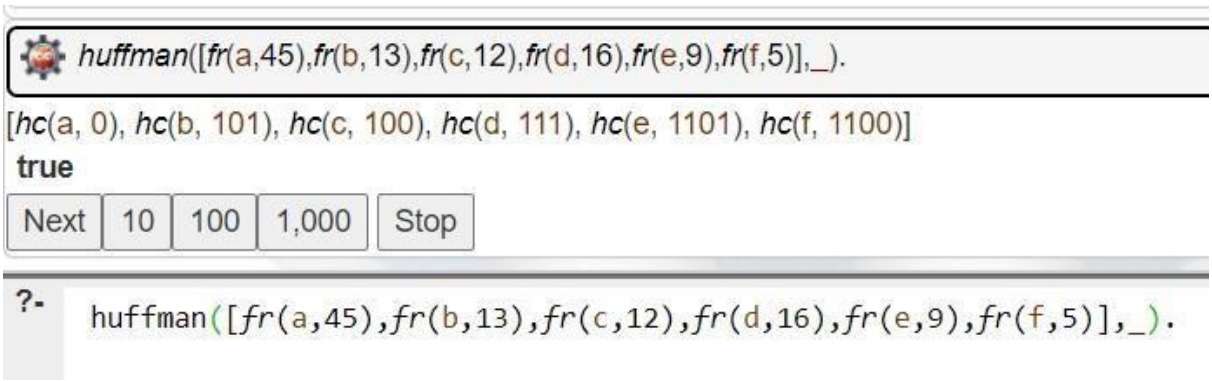
Q5) Huffman Code We suppose a set of symbols with their frequencies, given as a list of fr(S,F) terms. Example:
[fr(a,45),fr(b,13),fr(c,12),fr(d,16),fr(e,9),fr(f,5)]. Our objective is to construct a list hc(S,C) terms, where C is the Huffman code word for the symbol S. In our example, the result couldbe Hs =[hc(a,'0'), hc(b,'101'), hc(c,'100'), hc(d,'111'), hc(e,'1101'), hc(f,'1100')] [hc(a,'01'),...etc.]. The
task shall be performed by the predicate huffman/2 defined as follows: % huffman(Fs,Hs) :- Hs is the Huffman code table for the frequency table Fs

Code:

```
huffman(Fs,Cs) :-
  initialize(Fs,Ns),
  make_tree(Ns,T),
  traverse_tree(T,Cs).

initialize(Fs,Ns) :- init(Fs,NsU), sort(NsU,Ns).

init([],[]).
init([fr(S,F)|Fs],[n(F,S)|Ns]) :- init(Fs,Ns).

make_tree([T],T).
make_tree([n(F1,X1),n(F2,X2)|Ns],T) :-
  F is F1+F2,
  insert(n(F,s(n(F1,X1),n(F2,X2))),Ns,NsR),
  make_tree(NsR,T).

insert(N,[],[N]) :- !.insert(n(F,X),[n(F0,Y)|Ns],[n(F,X),n(F0,Y)|Ns]) :- F < F0, !.
insert(n(F,X),[n(F0,Y)|Ns],[n(F0,Y)|Ns1]) :- F >= F0, insert(n(F,X),Ns,Ns1).

traverse_tree(T,Cs) :- traverse_tree(T,'',Cs1-[]), sort(Cs1,Cs), write(Cs).
```

```
traverse_tree(n(_,A),Code,[hc(A,Code)|Cs]-Cs) :-
atom(A).traverse_tree(n(_,s(Left,Right)),Code,Cs1-Cs3) :-
  atom_concat(Code,'0',CodeLeft),
  atom_concat(Code,'1',CodeRight),
  traverse_tree(Left,CodeLeft,Cs1-Cs2),
  traverse_tree(Right,CodeRight,Cs2-
  Cs3).
```

Output:



Conclusion:

In this experiment, the aim was to learn prolog programming and to perform few tasks using it. Prolog is used for logic programming, it helps to solve logic based questions easily as in prolog we first set the facts and rules given in the question and then according to the rules and conditions we perform tasks and queries. Here I performed list operations using prolog such as removing duplicates from list, slicing a list, creating subsets of list of given size and also to generate huffman code for the given symbols and their respective frequencies. Also I used prolog to generate a family tree and wrote the required facts and rules and then executed the queries.